4 Advanced Counting Techniques

4.1 Mathematical induction (weak induction)

This section presents another proof technique (besides direct, contrapositive and contradiction)

1. mathematical induction is used to prove statements that are true for all positive integers or for positive integers greater than some number n_0
2. the proof of a statement $P(n), n \geq 1$ has two steps:
   (a) **Basis Step**: show that $P(1)$ is true The basis step is the foundation of the proof. For example, a statement can ask to prove that: for all positive integers we have that $n! \leq n^n$, and so the proof will start with:
   Proof: Show $P(1)$ is true: Note that $1! = 1$ is less than or equal to $1^1 = 1$. The inductive step will continue now.

   (b) **Inductive Step**: $\forall k \geq 1$ $(P(k) \rightarrow P(k+1))$ This step proves that every statement $P(k+1)$ is true given that the previous one, $P(k)$, is true. This implication is proved using a direct proof ($P(k)$ the inductive hypothesis).
3. So here is what the induction does: if $P(1)$ is T as we showed in the basis step, we can use the inductive step we obtain that $P(2)$ is T (here $k = 1$). And then, since we have that $P(2)$ is T, we use the inductive step to get that $P(3)$ is T. And since $P(3)$ is T, we again use the inductive step to get that $P(4)$ is T. And continuing in this manner, one may see how $P(n)$ is T by building up on the basis step and proving that we can go from any $P(k)$ to $P(k+1)$.
4. The basis step may not always be exactly $P(1)$. For example, if the results says to show that a statement is true for the positive integers greater than or equal to some number n_0, then $P(n_0)$ is what we have to show in the basis step ($P(1)$ is common, but it could be some $P(n_0)$ for $n_0 > 1$). For example, if the result states that: $\forall n > 2$ prove that $n! \leq n^n$, then the basis step will check $P(3)$: Note that $3! = 3 \cdot 2 \cdot 1 = 6$ is less than or equal to $3^3 = 27$.
5. types of problems that we will prove by mathematical induction:
   - summations
   - inequalities
   - divisibility

4.2 Recurrence Relations

1. This is a different version than the one in the book, which I hope it helps more as you look at the examples in both sections.
   Strong Induction: To prove that $P(n)$ is true for all positive integers $n \geq n_0$, where $P(n)$ is a propositional function, we complete two steps:

   - Basis step: verify that $P(n_0), P(n_0+1), \ldots, P(k_0)$ is true (that is verify the first case ($n = n_0$ which sometimes happens to be 1), and the other cases that cannot result from the inductive step, either because they don't follow the pattern or can't be used in the inductive step)
   - Inductive step: prove that $P(n_0) \wedge P(n_0+1) \wedge \ldots \wedge P(k)$ are true $\Rightarrow P(k+1)$ is true for all positive integers $k \geq k_0$.

2. to determine how many $P(n)$s you need to check in the basis step (i.e. to find k_0), one should first work on the inductive step to see what is the smallest value for k needed in the inductive step (or for which the inductive step works). One good example is Example 4 page 287, which presents a proof by induction, and then a proof by strong induction. In the inductive step of the proof by strong induction, when $P(k-3)$ is considered, we need $k - 3 \geq 12$ (since we're proving the result for $n \geq 12$). For this reason $k \geq 15$, and so $P(12), P(13), P(14), P(15)$ must be checked in the basis step. If this is not verified (i.e. only the first value is verified in the basis step of strong induction), then problems may occur– see problems 29 or 30 page 293.

3. it always helps to test a few cases for yourself before attempting to prove it, just to make sure that it works, and to see how to work the inductive step.

4. in practice, the weak form of mathematical induction (Section 4.1) should be the first attempted method of proof. If the proof is not straightforward, then strong induction might be better since it has more assumptions to be used in the proof.

5. skip the computational geometry examples

6. Well-Ordering Property: Every nonempty set of non-negative integers has a smallest element.

7. WOP is the basis for induction

8. one nice application is to prove relations involving recursively defined functions and sets (used in Dynamic programing for example)

4.3 Recursive Definitions and Structural Induction. (up to (not including) Structural induction)

1. Recursion: A process/formula that defines **recursively** the nth step based on (some) previous step/steps. The **basis step** needs to be defined in order to build up the formula from it.

2. recursively defined functions

   - Factorials (Recall that $F(n) = n! = n(n-1)(n-2)\cdots 3 \cdot 2 \cdot 1$):
     BASIS STEP: $F(0) = 1$
     RECURSIVE STEP:$F(n) = n \cdot F(n-1)$

   - Fibonacci numbers:
     BASIS STEP: $f_0 = 0$, and $f_1 = 1$,
     RECURSIVE STEP: $f_n = f_{n-1} + f_{n-2}, n \geq 2$.

3. well defined: there is a unique output for each input

4. recursively defined sets

   - Σ^* is the set of strings of the alphabet Σ, such that
     BASIS STEP: $\lambda \in \Sigma^*$ is the empty string (string containing no symbols)
     RECURSIVE STEP: if $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$,

   - Let Σ be a set of symbols and Σ^* be the set of strings formed from symbols in Σ. We define the concatination of two strings denoted by $\cdot$ as follows:
     BASIS STEP: if $w \in \Sigma^*$, then $w \cdot \lambda = w$, where λ is the empty string
     RECURSIVE STEP: if $w_1, w_2 \in \Sigma^*$ and $x \in \Sigma$, then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$

   - rooted trees consist of a set of vertices with a fixed vertex called root, and edges are defined by
     BASIS STEP: a single vertex is a degenerated rooted tree
     RECURSIVE STEP: T_{n+1} is formed from n rooted trees T_i with root r_i, by adding a new root r together with the edges rr_i $(1 \leq i \leq n)$.

     - extended binary trees $i = 2$ in the above definition, and T_1 or T_2 could be empty
     - full binary trees $i = 2$ in the above definition, and neither T_1 nor T_2 can be empty

5. up to (not including) Structural Induction